Interactive Design Exploration for Constrained Meshes

Bailin Deng*, Sofien Bouaziz, Mario Deuss, Alexandre Kaspar,
Yuliy Schwartzburg, Mark Pauly

Computer Graphics and Geometry Laboratory, EPFL, CH-1015 Lausanne, Switzerland

Abstract

In architectural design, surface shapes are commonly subject to geometric constraints imposed by material, fabrication or assembly. Rationalization algorithms can convert a freeform design into a form feasible for production, but often require design modifications that might not comply with the design intent. In addition, they only offer limited support for exploring alternative feasible shapes, due to the high complexity of the optimization algorithm.

We address these shortcomings and present a computational framework for interactive shape exploration of discrete geometric structures in the context of freeform architectural design. Our method is formulated as a mesh optimization subject to shape constraints. Our formulation can enforce soft constraints and hard constraints at the same time, and handles equality constraints and inequality constraints in a unified way. We propose a novel numerical solver that splits the optimization into a sequence of simple subproblems that can be solved efficiently and accurately.

Based on this algorithm, we develop a system that allows the user to explore designs satisfying geometric constraints. Our system offers full control over the exploration process, by providing direct access to the specification of the design space. At the same time, the complexity of the underlying optimization is hidden from the user, who communicates with the system through intuitive interfaces.

Keywords: Architectural geometry, Design exploration, Fabrication-aware design, Constraint-based modeling

1. Introduction

Digital tools have become ubiquitous in the architectural design process. For freeform architecture in particular, proper mathematical models, efficient geometry processing algorithms, and interactive shape editing software are essential for effective design. These tools provide great flexibility in creating complex

*Corresponding author. *Telephone*: +41 21 69 37533. *Fax*: +41 21 69 37540.
Email address: bailin.deng@epfl.ch (Bailin Deng)

architectural designs, but offer limited support for incorporating constraints imposed by material, fabrication, or assembly. One example is building with planar quadrilateral panels, which are popular for cost-effective realization of freeform structures with glass panels. In such a construction, the distance between the two diagonals of a panel needs to be smaller than a certain threshold to avoid large internal bending stress [1, 2]. These constraints are difficult to control manually, and typically require a separate rationalization process that maps the design to physical production. Rationalization needs to negotiate between physical constraints and design intent, often triggering several iterations to enable real-world fabrication of the digital design. This time-consuming process can lead to suboptimal designs or a significant increase in overall cost.

In this paper, we propose a new approach to geometric form finding and design that takes the constraints into account. Given a set of soft constraints and hard constraints, our approach enables the user to explore the space of shapes that satisfy the hard constraints exactly, and the soft constraints as much as possible. By integrating constraints into the design process, our approach yields designs that can be more effectively rationalized with respect to a given construction approach, thus avoiding unnecessary design iterations or suboptimal design solutions.

In our system, a typical design session proceeds as follows (Figure 1): First, an initial specification of constraints is provided by the designer. Starting from some initial shape, the user can then freely navigate feasible shapes by directly interacting with the current design. Our optimization algorithm computes a new design that stays within the constraint space, thus satisfying the geometric requirements imposed by the design rationale. This new design provides realtime visual feedback according to the user input, enabling effective exploration of design alternatives. The user can also alter the shape space by introducing new constraints, or by modifying or removing existing constraints. The design is automatically updated to remain in the new shape space. Such flexibility allows the user to test different rationalization options easily.

1.1. Related work

Computational methods for architectural design have become increasingly popular in recent years [3, 4]. With a focus on physical production, various rationalization algorithms have been proposed for many geometric goals such as: planar mesh optimization [5, 6, 7, 8, 9, 10, 11], multi-layer structures [12], single-curved panels [13], straight panels [14, 15], double-curved panels [16, 17], ruled panels [18, 19], circle and sphere packings [20], circular arc structures [21], point-folding structures [22] and functional webs [23]. These methods typically take a given freeform surface as input, and compute a surface decomposition that relates to a physical layout of panels. Usually such rationalization methods allow some deviation from the input reference surface to improve the quality of the paneling. They do not, however, support interactive exploration of the design alternatives.

Integrating rationalization methods into existing shape editing tools is typically not a viable option, since these algorithms often require minutes or some-

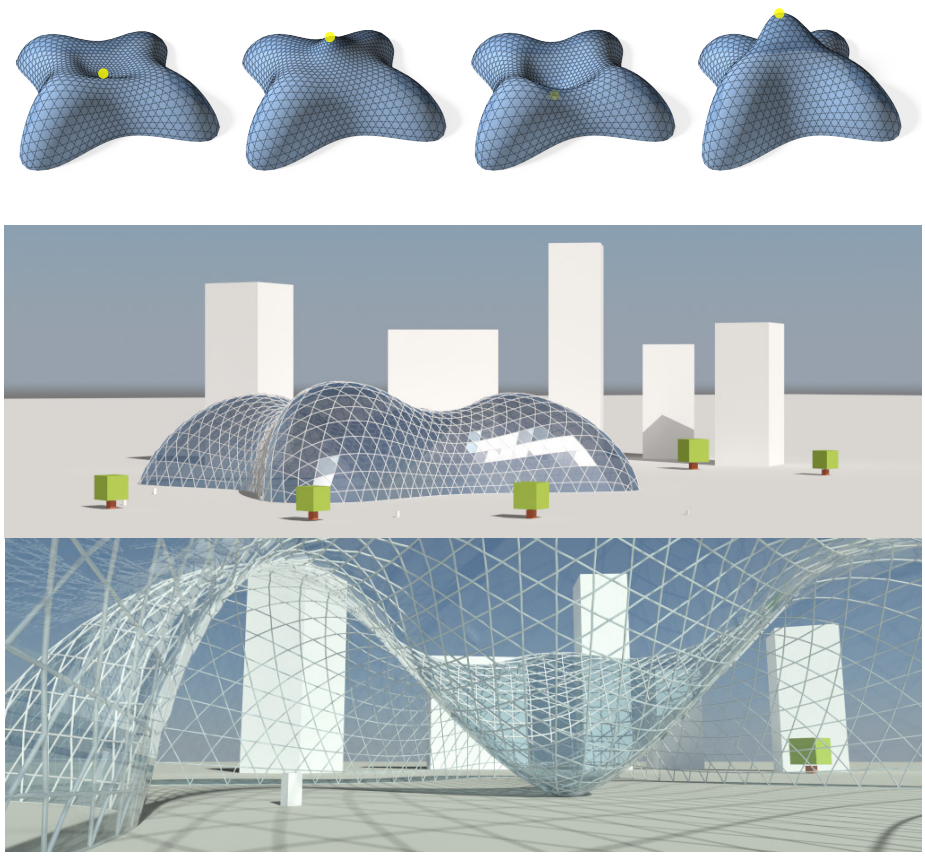

Figure 1: Handle-based constrained deformation of the Lilium tower, under hard constraints of planar faces (for all faces with more than three vertices) as well as soft constraints of regular polygonal faces (for all faces). Top: different feasible shapes during the exploration, with all boundary vertices and a set of interior vertices in the middle selected as handles. The interior vertex handles are moved during the exploration (shown in yellow). Middle and bottom: architectural design based on one of the feasible shapes.

times hours to compute a solution, thus preventing an interactive shape exploration process. Therefore, research efforts have focused on alternative approaches.

Geometric shape spaces have recently become popular as a tool for design exploration. One common interpretation of a shape space is a restriction of the space of all free parameters of a design. For example, Kilian et al. [24] represented a triangle mesh as a point in a high-dimensional space that treats each vertex coordinate as a free variable. They defined suitable Riemannian metrics on this space to restrict the embedding of the mesh to nearly isometric deformations of a given input surface. This allows exploring the manifold of shapes that approximately preserve lengths.

The method by Yang et al. [2] introduced a shape exploration tool for constrained meshes. They compute a local approximant of a high-dimensional constrained shape manifold, which the user can navigate efficiently. Based on this work, Zhao et al. [25] developed a guided exploration tool for the local approximant by automatically sampling new shapes. For these approaches, the generated shapes only satisfy the constraints approximately, and the constraint violation may exceed the tolerance for large deformations. Thus it is usually necessary to project the new mesh onto the constrained shape manifold to obtain feasible results. Compared to these methods, our system directly computes new shapes that satisfy all hard constraints, without the need for projection.

Vaxman [26] introduced a method for deforming polyhedral meshes while preserving the planarity of faces. In this method, a new mesh is computed by applying affine transformations to the faces of the old mesh. Poranne et al. [27] provided a characterization of maximal linear subspaces within the manifold of polyhedral surfaces, allowing shape exploration in a larger space than [26]. For meshes with general shape constraints, Bouaziz et al. [28] proposed a framework for imposing constraints to mesh elements. To solve the corresponding least squares optimization problem, they employ an alternating minimization approach to decompose the original problem into a sequence of simple subproblems. Similarly, Poranne et al. [29] combined alternating least squares and penalty method to planarize polygonal meshes. Both [28] and [29] enforce shape constraints using penalty methods, so they handle soft constraints only. Deng et al. [30] presented a constrained optimization approach to compute and explore local modifications for constrained meshes. Using an augmented Lagrangian solver, they computed new shapes that satisfy all constraints exactly, enabling the method to enforce hard constraints.

1.2. Overview and contribution

Our work deals with similar problems as [28], namely optimization and deformation of meshes subject to shape constraints. Unlike [28] and [29] which enforce soft constraints only, and [30] which only handles hard constraints, we allow both soft constraints and hard constraints at the same time, by unifying the approaches in [28] and [30]. The new mesh shape is computed by constrained optimization, where the soft constraints contribute to the objective function. By introducing auxiliary variables, we convert the problem into one with separable objective function and linear side constraints, which is solved using an augmented Lagrangian solver. The special structure of the problem allows us to decompose the solver into a set of simple subproblems. Each subproblem is either evaluating a proximal operator [31, 32], or solving a sparse symmetric positive definite linear system, both of which can be done efficiently. We implement the method on GPU to gain significant speedup from parallelism, which allows interactive exploration of the shape space. Our contribution includes:

- a unified formulation of equivalent conditions for shape constraints, equipped with proximal operators with intuitive geometric meanings;

- a general shape optimization framework that enforces both soft constraints and hard constraints, with an augmented Lagrangian solver consisting of simple subproblems that can be efficiently solved. Our solver outperforms the commonly used interior point method for interactive applications.

The paper is organized as follows. Section 2 presents the formulation of the optimization problem. Section 3 provides an efficient numerical solver, based on which a constrained shape exploration system is proposed in Section 4. Examples are given in Section 5 for design exploration using this system, followed by further discussion and conclusion in Section 6.

2. Formulation

2.1. Constraints for polygonal meshes

We focus in this work on polygonal meshes as construction-aware representations for freeform architectural surfaces. For example, such a mesh represents panels as faces [5], or represents curve elements as edge polylines [23]. Typically, the mesh is subject to a set of shape constraints related to fabrication or aesthetics. For a mesh with n_v vertices $\{v_i \mid i = 1, \ldots, n_v\}$ and *fixed connectivity*, we represent its shape using a vector $\mathbf{p} = [\mathbf{p}_1^T, \ldots, \mathbf{p}_{n_v}^T]^T \in \mathbb{R}^{3n_v}$, where $\mathbf{p}_i \in \mathbb{R}^3$ is the position of vertex v_i. In this paper, we consider constraints that can be represented as equality or inequality conditions about the vertex positions. Namely, each constraint has one of the following forms

$$\begin{array}{ll} \text{Equality constraint:} & \mathbf{f}(\mathbf{p}) = \mathbf{0}, \\ \text{Inequality constraint:} & \mathbf{a} \leq \mathbf{g}(\mathbf{p}) \leq \mathbf{b}, \end{array}$$

where $\mathbf{f}$, $\mathbf{g}$ are vector-valued functions. A constraint involving m vertices $v_{i_1}, \ldots, v_{i_m}$, whether equality or inequality, can always be represented as

$$\mathbf{M}\mathbf{p} \in \mathcal{C}.$$

Here matrix $\mathbf{M}$ collects all the involved vertex positions into a vector

$$\mathbf{M}\mathbf{p} = [\mathbf{p}_{i_1}^T, \ldots, \mathbf{p}_{i_m}^T]^T, \tag{1}$$

and $\mathcal{C}$ is the feasible set for this vector. If a constraint is *translation-invariant* (i.e., applying a common translation to all involved vertices does not change the status of constraint satisfaction), it can also be represented using one of the following forms of $\mathbf{M}$:

$$\mathbf{M}\mathbf{p} = \left[(\mathbf{p}_{i_1} - \frac{1}{m}\sum\nolimits_{k=1}^{m} \mathbf{p}_{i_k})^T, \ldots, (\mathbf{p}_{i_m} - \frac{1}{m}\sum\nolimits_{k=1}^{m} \mathbf{p}_{i_k})^T \right]^T, \tag{2}$$

$$\text{or} \quad \mathbf{M}\mathbf{p} = \left[(\mathbf{p}_{i_2} - \mathbf{p}_{i_1})^T, \ldots, (\mathbf{p}_{i_m} - \mathbf{p}_{i_1})^T \right]^T. \tag{3}$$

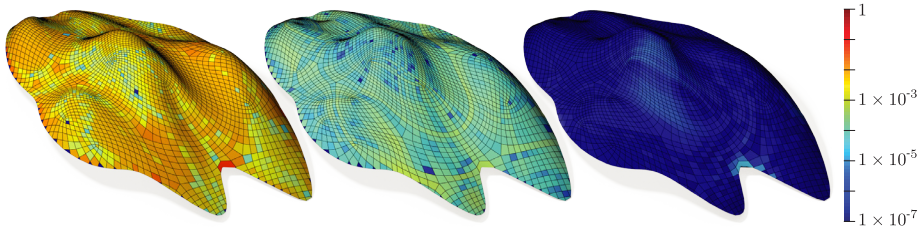

Figure 2: Planarization of a polygonal mesh using soft and hard constraints face planarity. Left: the initial mesh with an average edge length of 1.5. The boundary vertices are selected as handles, with target positions being the same as their initial positions. Middle: result with soft constraints of planar faces. Right: result with hard constraints of planar faces. Color-coding shows the maximum distance between a vertex and the least squares fitting plane for a face. The hard constraint result satisfies the constraints much better than the soft constraint result.

Remark. Forms (2) and (3) for constraint representation were used in [28] and [30] respectively. Compared to the "näive" form (1), they usually lead to faster convergence of alternating minimization solvers (see [28] for a detailed discussion).

Depending on the importance of each constraint, a designer may want to strictly enforce some constraints, while allowing others to be slightly violated. Therefore, we distinguish two types of constraints:

- *Hard constraints* are conditions that need to be satisfied exactly. For example, panel shapes may need to satisfy the constraints imposed by the chosen fabrication technology and construction material.
- *Soft constraints* are criteria that we would like the mesh to meet, but they are allowed to be violated if they conflict with other conditions. For example, the designer may want mesh faces to be regular polygons, in order to achieve more aesthetic shapes. But this condition can be relaxed for the mesh to have more degrees of freedom in its shape.

2.2. Optimization problem for constrained mesh deformation

Assume that we are given N_h hard constraints $\{\mathbf{M}_\mathrm{h}^i\mathbf{p} \in \mathcal{H}_i \mid i = 1, \ldots, N_\mathrm{h}\}$ and N_s soft constraints $\{\mathbf{M}_\mathrm{s}^j\mathbf{p} \in \mathcal{S}_j \mid j = 1, \ldots, N_\mathrm{s}\}$ for the mesh shape. Starting from an initial mesh, we provide a system where the user can explore other feasible mesh shapes with the same connectivity. In this system, the user first chooses a set of vertices as handles, and specifies their target positions in the new shape. A new mesh is computed based on the following criteria:

- The handle vertices are close to their target positions.
- The new mesh has a small value of fairness energy.
- All hard constraints are satisfied exactly.
- The soft constraints are satisfied as much as possible.

- The non-handle vertices are close to their initial positions.

The last condition above prevents unnecessary shape changes, and ensures that the problem is well-defined. Based on these criteria, the new mesh is computed by solving a constrained optimization problem

$$\begin{aligned} \min_{\mathbf{p}} \quad & w_{\mathrm{h}} F_{\mathrm{handle}} + w_{\mathrm{c}} F_{\mathrm{close}} + w_{\mathrm{f}} F_{\mathrm{fair}} + \sum_{j=1}^{N_{\mathrm{s}}} w_j^{\mathrm{s}} F_{\mathrm{soft}}^{(j)} \\ \text{s.t.} \quad & \mathbf{M}_{\mathrm{h}}^j \mathbf{p} \in \mathcal{H}_j, \qquad j = 1, \ldots, N_{\mathrm{h}}. \end{aligned} \tag{4}$$

Here w_{h}, w_{c}, w_{f} and w_j^{s} are positive weights. Functions F_{handle}, F_{close}, F_{fair} measure respectively the distance from handle vertices to their target positions, the distance from non-handle vertices to their original positions, and the fairness of the new mesh, respectively

$$F_{\mathrm{handle}} = \sum_{i \in \Gamma} \|\mathbf{p}_i - \mathbf{t}_i\|_2^2, \quad F_{\mathrm{close}} = \sum_{j \notin \Gamma} \|\mathbf{p}_j - \mathbf{p}_j^0\|_2^2, \quad F_{\mathrm{fair}} = \|\mathbf{L}(\mathbf{p} - \mathbf{p}^0)\|_2^2.$$

Here Γ is the index set for handle vertices, $\mathbf{t}_i$ is the target position for vertex v_i, $\mathbf{p}_j^0$ is the original position for vertex v_j, and vector $\mathbf{p}^0$ packs the original positions for all vertices. Matrix $\mathbf{L}$ measures the smoothness of vertex displacements $\mathbf{p} - \mathbf{p}^0$ across the mesh. It can be a Laplacian matrix, or a matrix that computes the second/third order difference terms along mesh polylines [2]. Function $F_{\mathrm{soft}}^{(j)}$ penalizes the violation of soft constraints $\mathbf{M}_{\mathrm{s}}^j \mathbf{p} \in \mathcal{S}_j$, which we define as the squared Euclidean distance between $\mathbf{M}_{\mathrm{s}}^j \mathbf{p}$ and $\mathcal{S}_j$

$$F_{\mathrm{soft}}^{(j)} = \left[\mathrm{dist}(\mathbf{M}_{\mathrm{s}}^j \mathbf{p}, \mathcal{S}_j)\right]^2.$$

3. Numerical solution

This optimization problem (4) usually involves a large number of nonconvex nonlinear constraints, and is difficult to solve efficiently. To achieve interactive results, we employ a strategy similar to [30]: First we introduce a set of auxiliary variables with additional constraints to convert (4) into a problem with separable target function and linear side constraints; the converted problem is then solved using an augmented Lagrangian method [33]. Similar to [30], the special structure of this problem enables us to decompose the solver into a set of simple subproblems, which can be solved in parallel to gain significant speedup. This section explains our method in detail.

3.1. Converted problem

First for each soft constraint $\mathbf{M}_{\mathrm{s}}^j \mathbf{p} \in \mathcal{S}_j$, we introduce auxiliary variables $\mathbf{s}_j \in \mathcal{S}_j$ to write its violation measure as

$$F_{\mathrm{soft}}^{(j)} = \min_{\mathbf{s}_j \in \mathcal{S}_j} \|\mathbf{M}_{\mathrm{s}}^j \mathbf{p} - \mathbf{s}_j\|_2^2.$$

Next for each hard constraint $\mathbf{M}_{\mathrm{h}}^i\mathbf{p} \in \mathcal{H}_i$, we introduce auxiliary variables $\mathbf{h}_i \in \mathcal{H}_i$ to rewrite the condition as $\mathbf{M}_{\mathrm{h}}^i\mathbf{p} = \mathbf{h}_i$. Using the function definitions in Section 2, and incorporating new variables $\mathbf{h} = [\mathbf{h}_1^T, \ldots, \mathbf{h}_{N_{\mathrm{h}}}^T]^T$ and $\mathbf{s} = [\mathbf{s}_1^T, \ldots, \mathbf{s}_{N_{\mathrm{s}}}^T]^T$, we derive the following equivalent problem

$$\begin{aligned} \min_{\mathbf{p},\mathbf{h},\mathbf{s}} \quad & \frac{1}{2}\|\mathbf{D}\mathbf{p} - \mathbf{r}\|_2^2 + \frac{w_{\mathrm{f}}}{2}\|\mathbf{L}(\mathbf{p} - \mathbf{p}^0)\|_2^2 + \sum_{j=1}^{N_{\mathrm{s}}} \frac{w_j^{\mathrm{s}}}{2}\|\mathbf{M}_{\mathrm{s}}^j\mathbf{p} - \mathbf{s}_j\|_2^2 \\ & + \sum_{j=1}^{N_{\mathrm{s}}} \sigma_{\mathcal{S}_j}(\mathbf{s}_j) + \sum_{i=1}^{N_{\mathrm{h}}} \sigma_{\mathcal{H}_i}(\mathbf{h}_i) \\ \text{s.t.} \quad & \mathbf{M}_{\mathrm{h}}^i\mathbf{p} = \mathbf{h}_i, \quad i = 1, \ldots, N_{\mathrm{h}}, \end{aligned} \tag{5}$$

where

$$\mathbf{D} = \begin{bmatrix} d_1\mathbf{I}_3 & & \\ & \ddots & \\ & & d_{n_v}\mathbf{I}_3 \end{bmatrix}, \qquad \mathbf{r} = \begin{bmatrix} \mathbf{r}_1 \\ \vdots \\ \mathbf{r}_{n_v} \end{bmatrix},$$

with $\mathbf{I}_3$ being the 3×3 identity matrix, and

$$d_i = \begin{cases} \sqrt{w_{\mathrm{h}}} & \text{if } i \in \Gamma \\ \sqrt{w_{\mathrm{c}}} & \text{otherwise} \end{cases}, \quad \mathbf{r}_i = \begin{cases} d_i\mathbf{t}_i & \text{if } i \in \Gamma \\ d_i\mathbf{p}_i^0 & \text{otherwise} \end{cases}, \quad \text{for } i = 1, \ldots, n_v.$$

Indicator functions $\sigma_{\mathcal{S}_j}(\mathbf{s}_j)$ and $\sigma_{\mathcal{H}_i}(\mathbf{h}_i)$ ensure $\mathbf{s}_j \in \mathcal{S}_j$ and $\mathbf{h}_i \in \mathcal{H}_i$, respectively

$$\sigma_{\mathcal{S}_j}(\mathbf{s}_j) = \begin{cases} 0 & \text{if } \mathbf{s}_j \in \mathcal{S}_j \\ +\infty & \text{otherwise} \end{cases}, \qquad \sigma_{\mathcal{H}_i}(\mathbf{h}_i) = \begin{cases} 0 & \text{if } \mathbf{h}_i \in \mathcal{H}_i \\ +\infty & \text{otherwise} \end{cases}.$$

Remark. In this formulation, auxiliary variables from different constraints are independent from each other. If a vertex is involved in multiple constraints, it will induce different auxiliary variables in each constraint. As a result, there can be a large number of auxiliary variables. This seemingly redundant formulation is actually the key to high efficiency, as it splits the target function into separable terms that can be handled in parallel (see the next section for details).

3.2. Augmented Lagrangian solver

In optimization problem (5), let the target function be denoted by $F(\mathbf{p}, \mathbf{h}, \mathbf{s})$, and the violation of side constraints be denoted by $\mathbf{c}_i(\mathbf{p}, \mathbf{h}) = \mathbf{M}_{\mathrm{h}}^i\mathbf{p} - \mathbf{h}_i$. The augmented Lagrangian method solves (5) by searching for a saddle point of the *augmented Lagrangian function*

$$\mathcal{L}(\mathbf{p}, \mathbf{h}, \mathbf{s}, \boldsymbol{\lambda}; \mu) = F(\mathbf{p}, \mathbf{h}, \mathbf{s}) + \sum_{i=1}^{N_{\mathrm{h}}} \left[\boldsymbol{\lambda}_i^T\mathbf{c}_i(\mathbf{p}, \mathbf{h}) + \frac{\mu}{2}\|\mathbf{c}_i(\mathbf{p}, \mathbf{h})\|_2^2\right].$$

Here $\boldsymbol{\lambda}_i$ are Lagrangian multiplier vectors, which together form the *dual variable* $\boldsymbol{\lambda} = \left[\boldsymbol{\lambda}_1^T, \ldots, \boldsymbol{\lambda}_{N_{\mathrm{h}}}^T\right]^T$. $\mathbf{p}, \mathbf{h}$ and $\mathbf{s}$ are called the *primal variables*, and $\mu > 0$ is a *penalty parameter*. Our solver iteratively updates $\mathbf{p}, \mathbf{h}, \mathbf{s}, \boldsymbol{\lambda}$ and μ until convergence. In each iteration, new values $(\widehat{\mathbf{p}}, \widehat{\mathbf{h}}, \widehat{\mathbf{s}}, \widehat{\boldsymbol{\lambda}}, \widehat{\mu})$ are computed from current values $(\overline{\mathbf{p}}, \overline{\mathbf{h}}, \overline{\mathbf{s}}, \overline{\boldsymbol{\lambda}}, \overline{\mu})$ using the following steps:

1. *Primal update*: $(\widehat{\mathbf{p}}, \widehat{\mathbf{h}}, \widehat{\mathbf{s}}) = \underset{\mathbf{p},\mathbf{h},\mathbf{s}}{\operatorname{argmin}} \mathcal{L}(\mathbf{p}, \mathbf{h}, \mathbf{s}, \overline{\boldsymbol{\lambda}}; \overline{\mu})$.
2. *Dual update*: $\widehat{\boldsymbol{\lambda}}_i = \overline{\boldsymbol{\lambda}}_i + \overline{\mu}\ \mathbf{c}_i(\widehat{\mathbf{p}}, \widehat{\mathbf{h}})$, $i = 1, \ldots, N_{\mathrm{h}}$.
3. *Penalty update*: choose $\widehat{\mu} \geq \overline{\mu}$.

The primal update step is explained below. For the penalty update step and the convergence criteria, refer to [30] for more details.

3.2.1. Primal update

The primal update minimizes a function of primal variables $\widehat{\mathcal{L}}(\mathbf{p}, \mathbf{h}, \mathbf{s}) = \mathcal{L}(\mathbf{p}, \mathbf{h}, \mathbf{s}, \overline{\lambda}; \overline{\mu})$. For this we employ an alternating minimization strategy: Starting from initial values $\mathbf{p}^{(0)} = \overline{\mathbf{p}}$, $\mathbf{h}^{(0)} = \overline{\mathbf{h}}$, $\mathbf{s}^{(0)} = \overline{\mathbf{s}}$, we iteratively perform the following updates until convergence:

1. Fix $\mathbf{p}$, update $\mathbf{h}$, $\mathbf{s}$: $\quad \left(\mathbf{h}^{(k+1)}, \mathbf{s}^{(k+1)}\right) = \underset{\mathbf{h},\mathbf{s}}{\operatorname{argmin}} \widehat{\mathcal{L}}\left(\mathbf{p}^{(k)}, \mathbf{h}, \mathbf{s}\right)$.
2. Fix $\mathbf{h}$, $\mathbf{s}$, update $\mathbf{p}$: $\quad \mathbf{p}^{(k+1)} = \underset{\mathbf{p}}{\operatorname{argmin}} \widehat{\mathcal{L}}\left(\mathbf{p}, \mathbf{h}^{(k+1)}, \mathbf{s}^{(k+1)}\right)$.

$(\mathbf{h}, \mathbf{s})$-*update.* This problem is separable with respect to the auxiliary variables for each constraint. Thus we have independent subproblems that can be solved in parallel (superscript counters are ignored to simplify notations),

$$\min_{\mathbf{h}_i} \|\mathbf{h}_i - (\mathbf{M}_{\mathrm{h}}^i \mathbf{p} + \frac{\boldsymbol{\lambda}_i}{\mu})\|_2^2 + \sigma_{\mathcal{H}_i}(\mathbf{h}_i), \quad i = 1, \ldots, N_{\mathrm{h}}, \tag{6}$$

$$\min_{\mathbf{s}_j} \|\mathbf{s}_j - \mathbf{M}_{\mathrm{s}}^j \mathbf{p}\|_2^2 + \sigma_{\mathcal{S}_j}(\mathbf{s}_j). \quad j = 1, \ldots, N_{\mathrm{s}}. \tag{7}$$

Geometrically, the solutions are the Euclidean projections of $\mathbf{M}_{\mathrm{h}}^i \mathbf{p} + \boldsymbol{\lambda}_k/\mu$ and $\mathbf{M}_{\mathrm{s}}^j \mathbf{p}$ onto feasible sets $\mathcal{H}_i$ and $\mathcal{S}_j$ respectively, which can be computed efficiently for many shape constraints (see Section 3.3).

$\mathbf{p}$-*update.* This is equivalent to

$$\min_{\mathbf{p}} \ \|\mathbf{D}\mathbf{p} - \mathbf{r}\|_2^2 + w_{\mathrm{f}} \|\mathbf{L}(\mathbf{p} - \mathbf{p}^0)\|_2^2 + \sum_{j=1}^{N_{\mathrm{s}}} w_j^{\mathrm{s}} \|\mathbf{M}_{\mathrm{s}}^j \mathbf{p} - \mathbf{s}_j\|_2^2 + \mu \sum_{i=1}^{N_{\mathrm{h}}} \|\mathbf{M}_{\mathrm{h}}^i \mathbf{p} - \mathbf{h}_i + \frac{\boldsymbol{\lambda}_i}{\mu}\|_2^2,$$

which reduces to solving a symmetric positive definite sparse linear system

$$\begin{aligned} & \left[\mathbf{D}^T\mathbf{D} + w_{\mathrm{f}}\mathbf{L}^T\mathbf{L} + \mu \sum_{i=1}^{N_{\mathrm{h}}} \left(\mathbf{M}_{\mathrm{h}}^i\right)^T \mathbf{M}_{\mathrm{h}}^i + \sum_{j=1}^{N_{\mathrm{s}}} w_j^{\mathrm{s}} \left(\mathbf{M}_{\mathrm{s}}^j\right)^T \mathbf{M}_{\mathrm{s}}^j\right] \mathbf{p} \\ = \quad & \mathbf{D}^T\mathbf{r} + w_{\mathrm{f}}\mathbf{L}^T\mathbf{L}\mathbf{p}^0 + \mu \sum_{i=1}^{N_{\mathrm{h}}} \left(\mathbf{M}_{\mathrm{h}}^i\right)^T \left(\mathbf{h}_i - \frac{\boldsymbol{\lambda}_i}{\mu}\right) + \sum_{j=1}^{N_{\mathrm{s}}} w_j^{\mathrm{s}} \left(\mathbf{M}_{\mathrm{s}}^j\right)^T \mathbf{s}_j. \end{aligned}$$

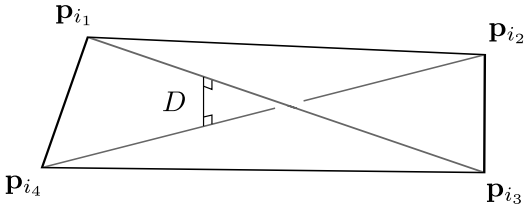

Figure 3: Distance D between the diagonals of a quad with vertices $\mathbf{p}_{i_1}, \mathbf{p}_{i_2}, \mathbf{p}_{i_3}, \mathbf{p}_{i_4}$.

3.3. Proximal operators

Our solver can be easily adapted to different types of shape constraints. The only difference is the solution to the primal update subproblem for auxiliary variables, which has a general form

$$\min_{\mathbf{y}} \|\mathbf{y} - \mathbf{x}\|_2^2 + \sigma_{\mathcal{C}}(\mathbf{y}). \tag{8}$$

Its solution is the so-called *proximal operator* for indicator function $\sigma_{\mathcal{C}}$ [32]. Geometrically, it is the Euclidean projection of $\mathbf{x}$ onto feasible set $\mathcal{C}$, which means moving $\mathbf{x}$ by the least amount to satisfy the constraint.

In Section 3.1, we introduce auxiliary variables to convert the original optimization problem. This is an instance of a numerical optimization strategy called *variable splitting* [34]. Proximal operators together with variable splitting play an important role in numerical optimization, since they provide simple separable reformulations of the optimization problems. Such reformulations can be easily parallelized, and enable efficient solutions to many problems that are otherwise difficult to solve [31, 32]. Our method is most useful when the proximal operators can be efficiently evaluated, which is indeed the case for many shape constraints relevant to architectural design and fabrication [28]. In the following sections, we present some proximal operators that are used in this paper.

3.3.1. Bounded diagonal distance of quad faces

When realizing freeform shapes using planar quadrilateral glass panels, a certain tolerance is usually allowed for the violation of planarity. Such violation is measured using the distance between the two diagonal lines of the quad, leading to the following fabrication constraint for each quad face with vertex positions $\mathbf{p}_{i_1}$, $\mathbf{p}_{i_2}$, $\mathbf{p}_{i_3}$, $\mathbf{p}_{i_4}$ [2] (see Figure 3)

$$D(\mathbf{p}_{i_1}, \mathbf{p}_{i_2}, \mathbf{p}_{i_3}, \mathbf{p}_{i_4}) \leq \epsilon, \tag{9}$$

where $D(\mathbf{p}_{i_1}, \mathbf{p}_{i_2}, \mathbf{p}_{i_3}, \mathbf{p}_{i_4})$ is the distance between diagonal lines $\overline{\mathbf{p}_{i_1}\mathbf{p}_{i_3}}$, $\overline{\mathbf{p}_{i_2}\mathbf{p}_{i_4}}$, and $\epsilon > 0$ is a tolerance value. Since this constraint is translation-invariant, we can represent it as in the form of (2)

$$[(\mathbf{p}_{i_1} - \mathbf{m}_{\mathbf{p}})^T, (\mathbf{p}_{i_2} - \mathbf{m}_{\mathbf{p}})^T, (\mathbf{p}_{i_3} - \mathbf{m}_{\mathbf{p}})^T, (\mathbf{p}_{i_4} - \mathbf{m}_{\mathbf{p}})^T]^T \in \mathcal{C},$$

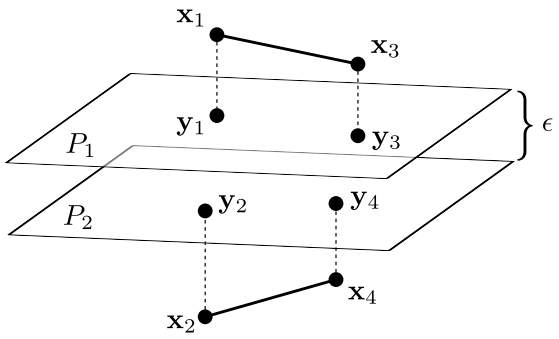

Figure 4: Proximal operator for bounded distance between diagonal lines.

where $\mathbf{m_p} = \frac{1}{4}\sum_{k=1} \mathbf{p}_{i_k}$, and $\mathcal{C} = \{\mathbf{q} = [\mathbf{q}_1^T, \mathbf{q}_2^T, \mathbf{q}_3^T, \mathbf{q}_4^T]^T \mid D(\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3, \mathbf{q}_4) \leq \epsilon\}$. The proximal operator problem has the form

$$\min_{\mathbf{y}} \sum_{k=1}^{4} \|\mathbf{y}_k - \mathbf{x}_k\|_2^2 + \sigma_{\mathcal{C}}(\mathbf{y}),$$

with auxiliary variable $\mathbf{y} = [\mathbf{y}_1^T, \mathbf{y}_2^T, \mathbf{y}_3^T, \mathbf{y}_4^T]^T$, and $\mathbf{x}_k, \mathbf{y}_k \in \mathbb{R}^3$ $(k = 1, 2, 3, 4)$. If $D(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) \leq \epsilon$, then we have a trivial solution $\mathbf{y}_k = \mathbf{x}_k$ for all k. Thus we consider only the nontrivial case where $D(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) > \epsilon$, namely the distance between diagonal lines of quadrilateral $\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4$ is larger than the tolerance ϵ. The solution is a closest quadrilateral for which the distance between diagonals is exactly ϵ; thus we only need to solve (see Figure 4)

$$\min_{\mathbf{y}} \sum_{k=1}^{4} \|\mathbf{y}_k - \mathbf{x}_k\|_2^2 \qquad \text{s.t.} \;\; D(\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \mathbf{y}_4) = \epsilon. \tag{10}$$

For the solution points $\mathbf{y}_1$, $\mathbf{y}_2$, $\mathbf{y}_3$, $\mathbf{y}_4$, there must exist two parallel planes P_1 and P_2 that contain $\mathbf{y}_1$, $\mathbf{y}_3$ and $\mathbf{y}_2$, $\mathbf{y}_4$ respectively, and are distance ϵ apart. These two planes share a common unit normal vector $\mathbf{n}$, and can be represented using the linear equations

$$\begin{aligned} P_1 &: \; \{\mathbf{z} \in \mathbb{R}^3 \mid \mathbf{z} \cdot \mathbf{n} = d\}, \\ P_2 &: \; \{\mathbf{z} \in \mathbb{R}^3 \mid \mathbf{z} \cdot \mathbf{n} = d + \epsilon\}, \end{aligned}$$

where d is a scalar that represents the signed distance from plane P_1 to the origin. Since $\mathbf{y}_k$ needs to be closest to $\mathbf{x}_k$, $\mathbf{y}_1$, $\mathbf{y}_3$ are the closest projections of $\mathbf{x}_1$, $\mathbf{x}_3$ onto P_1, and $\mathbf{y}_2$, $\mathbf{y}_4$ are the closest projections of $\mathbf{x}_2$, $\mathbf{x}_4$ onto P_2. Thus the target function in (10) can be written as a function h of $\mathbf{n}$ and d,

$$h = \sum_{k=1}^{4} \|\mathbf{x}_k - \mathbf{y}_k\|_2^2 = \sum_{i=1,3} (\mathbf{x}_i \cdot \mathbf{n} - d)^2 + \sum_{j=2,4} (\mathbf{x}_j \cdot \mathbf{n} - d - \epsilon)^2.$$

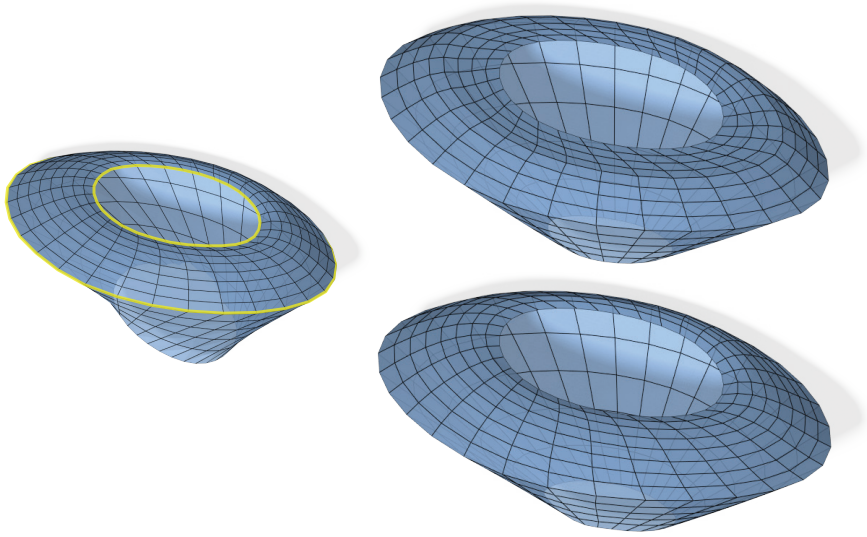

Figure 5: Comparison between the constraints of face planarity and bounded diagonal distance. Left: a given quad mesh whose faces are not planar. All vertices on the top boundary curve and the sharp circular polyline between the roof and the side (shown in yellow) are selected as handles, with target positions the same as the initial positions. Top right: result under hard constraints for face planarity for all faces. Bottom right: result under hard constraints of bounded diagonal distance for all faces. Bounded diagonal distance leads to more freedom in shapes than exact planarity, and results in smooth shapes in this case (note the kink in the top right figure).

From the optimality condition $\dfrac{\partial h}{\partial d} = 0$, we obtain

$$d = \mathbf{n} \cdot \mathbf{m}_{\mathbf{x}} - \frac{\epsilon}{2},$$

where $\mathbf{m}_{\mathbf{x}} = \dfrac{1}{4}\sum_{k=1}^{4} \mathbf{x}_k$. Substituting this into the definition of h, we obtain

$$h = \|\mathbf{X}\mathbf{n} - \mathbf{b}\|_2^2$$

where

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T - \mathbf{m}_{\mathbf{x}}^T \\ \mathbf{x}_2^T - \mathbf{m}_{\mathbf{x}}^T \\ \mathbf{x}_3^T - \mathbf{m}_{\mathbf{x}}^T \\ \mathbf{x}_4^T - \mathbf{m}_{\mathbf{x}}^T \end{bmatrix}, \qquad \mathbf{b} = \frac{1}{2}\begin{bmatrix} -\epsilon \\ \epsilon \\ -\epsilon \\ \epsilon \end{bmatrix}.$$

Then the unit normal vector $\mathbf{n}$ is the solution to

$$\min_{\mathbf{n}} \ \|\mathbf{X}\mathbf{n} - \mathbf{b}\|_2^2 \quad \text{s.t. } \|\mathbf{n}\|_2 = 1.$$

This is a least squares minimization over the unit sphere, which can be solved using the method in Section 12.1.2 of [35]. From the values of $\mathbf{n}$ and d, we

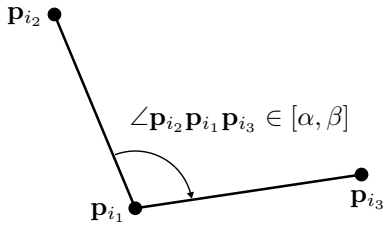

Figure 6: Bound constraint for angle $\angle \mathbf{p}_{i_2}\mathbf{p}_{i_1}\mathbf{p}_{i_3}$.

obtain $\mathbf{y}_k$ $(k = 1, 2, 3, 4)$ as

$$\mathbf{y}_k = \begin{cases} \mathbf{x}_k + \mathbf{n}\left[-\dfrac{\epsilon}{2} - (\mathbf{x}_k - \mathbf{m}_\mathbf{x}) \cdot \mathbf{n}\right], & k = 1, 3, \\ \mathbf{x}_k + \mathbf{n}\left[\dfrac{\epsilon}{2} - (\mathbf{x}_k - \mathbf{m}_\mathbf{x}) \cdot \mathbf{n}\right], & k = 2, 4. \end{cases}$$

3.3.2. Bounded angles

When using a polygonal mesh to represent realizations of freeform designs, we can represent the nodes, beams, and panels of the surface with the vertices, edges, and faces of the mesh, respectively. For such a mesh we can require the corner angles of each face to stay within a given range of values, in order to prevent small angles between adjacent beams and facilitate the fabrication of nodes and panels. Let $\alpha, \beta \in (0, \pi)$ be the minimum and maximum values that are allowed for interior angles. For the angle $\angle \mathbf{p}_{i_2}\mathbf{p}_{i_1}\mathbf{p}_{i_3}$ between edges $\mathbf{p}_{i_1}\mathbf{p}_{i_2}$ and $\mathbf{p}_{i_1}\mathbf{p}_{i_3}$, this constraint means (see Figure 6)

$$\cos\beta \leq \frac{(\mathbf{p}_{i_2} - \mathbf{p}_{i_1}) \cdot (\mathbf{p}_{i_3} - \mathbf{p}_{i_1})}{\|\mathbf{p}_{i_2} - \mathbf{p}_{i_1}\|_2 \, \|\mathbf{p}_{i_3} - \mathbf{p}_{i_1}\|_2} \leq \cos\alpha. \tag{11}$$

This translation-invariant constraint can be represented in the form of (3)

$$[(\mathbf{p}_{i_2} - \mathbf{p}_{i_1})^T, (\mathbf{p}_{i_3} - \mathbf{p}_{i_1})^T]^T \in \mathcal{C},$$

where

$$\mathcal{C} = \left\{ \mathbf{q} = [\mathbf{q}_1^T, \mathbf{q}_2^T]^T \;\middle|\; \mathbf{q}_1, \mathbf{q}_2 \in \mathbb{R}^3, \;\; \cos\beta \leq \frac{\mathbf{q}_1 \cdot \mathbf{q}_2}{\|\mathbf{q}_1\|_2 \, \|\mathbf{q}_2\|_2} \leq \cos\alpha \right\}.$$

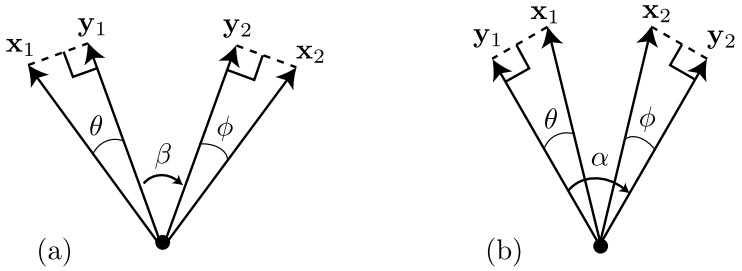

Figure 7: Proximal operator for bounded angles.

The proximal operator problem is

$$\min_{\mathbf{y}} \sum_{i=1,2} \|\mathbf{y}_i - \mathbf{x}_i\|_2^2 + \sigma_{\mathcal{C}}(\mathbf{y}),$$

with auxiliary variable $\mathbf{y} = [\mathbf{y}_1^T, \mathbf{y}_2^T]^T$, and $\mathbf{x}_1, \mathbf{x}_2, \mathbf{y}_1, \mathbf{y}_2 \in \mathbb{R}^3$. We only consider the non-trivial case where $[\mathbf{x}_1^T, \mathbf{x}_2^T]^T \notin \mathcal{C}$, i.e. the angle between $\mathbf{x}_1$ and $\mathbf{x}_2$ is outside the range $[\alpha, \beta]$. There are two possibilities:

1. If the angle between $\mathbf{x}_1$ and $\mathbf{x}_2$ is larger than β, the problem reduces to (see Figure 7(a))

$$\min_{\mathbf{y}_1, \mathbf{y}_2} \sum_{i=1,2} \|\mathbf{y}_i - \mathbf{x}_i\|_2^2 \qquad \text{s.t. } \frac{\mathbf{y}_1 \cdot \mathbf{y}_2}{\|\mathbf{y}_1\|_2 \ \|\mathbf{y}_2\|_2} = \cos\beta.$$

2. When the angle between $\mathbf{x}_1$, $\mathbf{x}_2$ is smaller than α, the problem becomes (see Figure 7(b))

$$\min_{\mathbf{y}_1, \mathbf{y}_2} \sum_{i=1,2} \|\mathbf{y}_i - \mathbf{x}_i\|_2^2 \qquad \text{s.t. } \frac{\mathbf{y}_1 \cdot \mathbf{y}_2}{\|\mathbf{y}_1\|_2 \ \|\mathbf{y}_2\|_2} = \cos\alpha.$$

Both cases require minimum displacement from $\mathbf{x}_1$ and $\mathbf{x}_2$ to reach a designated angle between them. To achieve minimum displacement, the solution $\mathbf{y}_1$, $\mathbf{y}_2$ must lie on the plane spanned by $\mathbf{x}_1$ and $\mathbf{x}_2$. Therefore, both cases reduce to 2D problems. In the first case $\mathbf{x}_1$ and $\mathbf{x}_2$ need to be moved towards each other, while in the second case they are moved away from each other. Let θ and ϕ be the angles between $\mathbf{x}_1$, $\mathbf{y}_1$ and between $\mathbf{x}_2$, $\mathbf{y}_2$, respectively. Let η be the sum of θ and ϕ, and let γ be the angle between $\mathbf{x}_1$ and $\mathbf{x}_2$. In the first case $\eta = \gamma - \beta$, while in the second case $\eta = \alpha - \gamma$. The minimum displacement from $\mathbf{x}_1$ to $\mathbf{y}_1$ implies that $\mathbf{y}_1 - \mathbf{x}_1$ must be orthogonal to $\mathbf{y}_1$, so

$$\|\mathbf{y}_1 - \mathbf{x}_1\|_2^2 = \left(\|\mathbf{x}_1\|_2 \sin\theta\right)^2.$$

Similarly, we have

$$\|\mathbf{y}_2 - \mathbf{x}_2\|_2^2 = \left(\|\mathbf{x}_2\|_2 \sin\phi\right)^2 = \left[\|\mathbf{x}_2\|_2 \sin(\eta - \theta)\right]^2.$$

Now the original constrained problem is reduced to an unconstrained one

$$\min_{\theta} \ \left(\|\mathbf{x}_1\|_2 \sin\theta\right)^2 + \left[\|\mathbf{x}_2\|_2 \sin(\eta - \theta)\right]^2.$$

The solution to this problem is

$$\theta = \frac{1}{2} \arctan \frac{\|\mathbf{x}_2\|_2^2 \sin(2\eta)}{\|\mathbf{x}_1\|_2^2 + \|\mathbf{x}_2\|_2^2 \cos(2\eta)}.$$

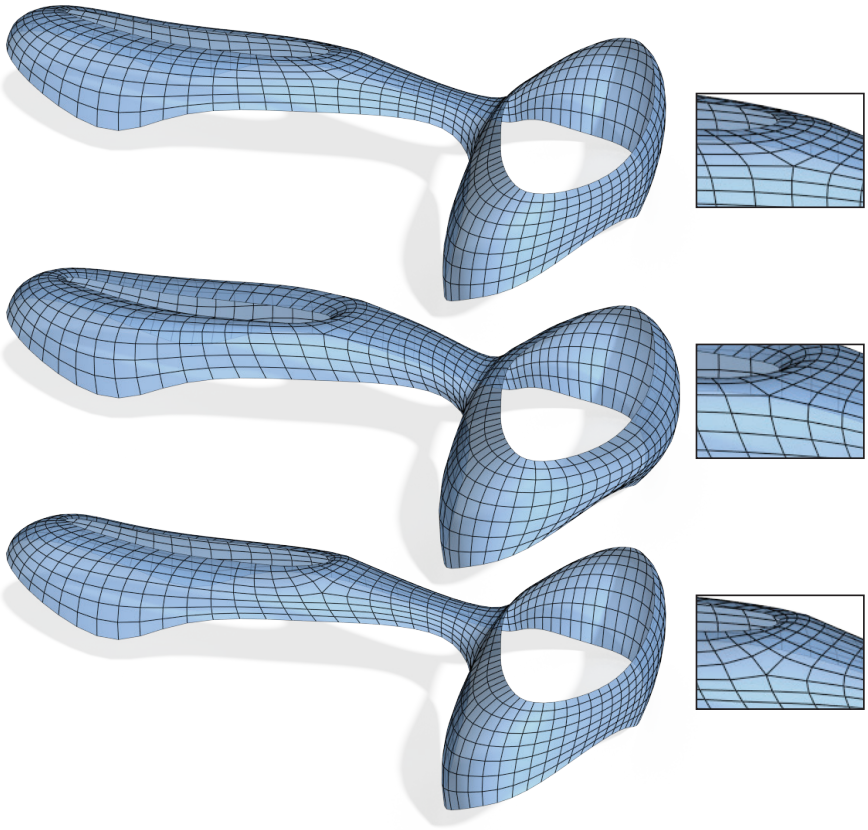

Figure 8: Switching between different shape spaces by changing constraint types. Top: an initial planar quad mesh. All boundary vertices are selected as handles, with target positions the same as their initial positions. Middle: result shape under hard constraints for face planarity and soft constraints for regularity of polygons. Note the shape changes of the quads in the middle part of the mesh due to the constraints of regular polygons. Bottom: a different result shape, by replacing the regular polygon constraints with hard constraints of bounded angles between 60 and 120 degrees. There is a notable change of face shapes around the singularity (see the zoomed-in images), where the initial mesh and the previous result mesh violate the angle constraints. On the other hand, the faces are thinner due to the lack of regular polygon constraints.

3.3.3. Other shape constraints

Below we list other shape constraints that appear in the examples in Section 5. Their proximal operators can be found in [28].

- *Planar faces*: All vertices of a face lie on a common plane.
- *Regular polygonal faces*: All edges of a face form a regular planar polygon.
- *Bounded edge length*: The length of each mesh edge stays within a given range $[l_{\min}, l_{\max}]$.

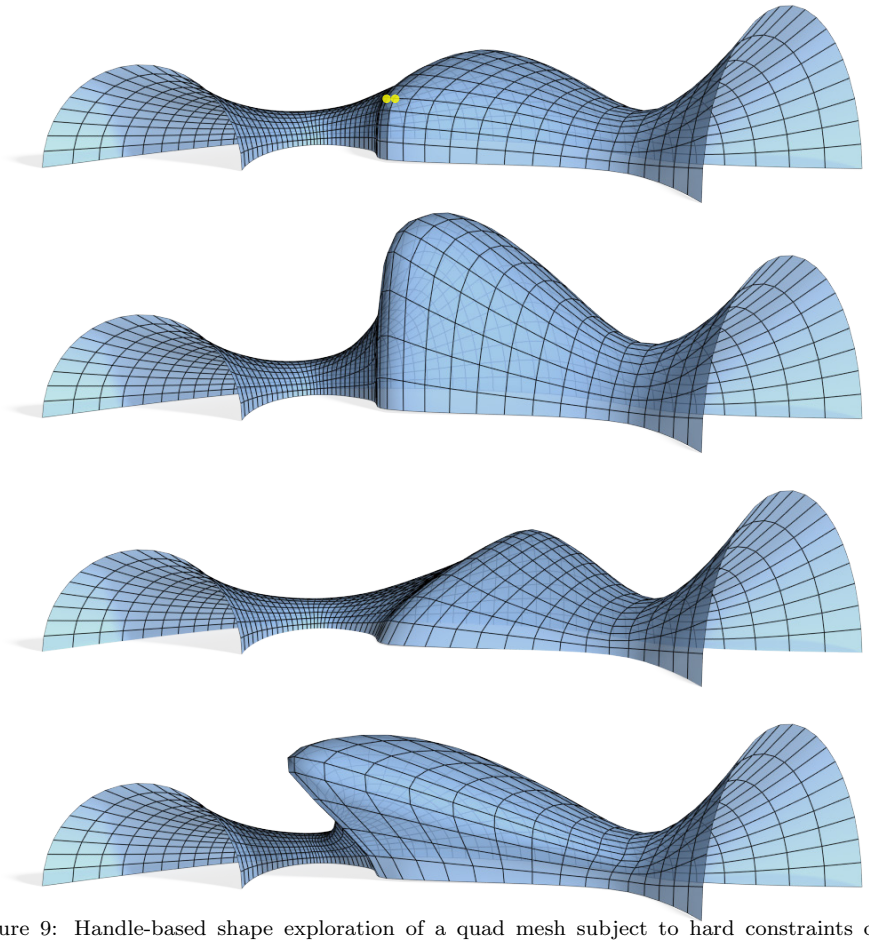

Figure 9: Handle-based shape exploration of a quad mesh subject to hard constraints of bounded diagonal distance of faces. The initial mesh (shown on the top) is a planar quad mesh. Two interior vertices (shown in yellow) and all boundary vertices are selected as handles. During the exploration only the interior handles are moved.

4. Interactive shape exploration

Using the algorithm in the previous section, we implement an interactive exploration system for mesh shapes subject to constraints. First in the specification phase, the user specifies the soft constraints and hard constraints for an initial mesh, and selects handle vertices. Then in the exploration phase, the mesh shape is updated by solving the optimization problem (5). When the user drags the handles, the mesh shape is consecutively updated using a sequence of handle positions obtained during the dragging, providing intuitive visual feedback to the user about feasible shapes and their relations to handle positions. Alternatively, the user can leave the handle vertices at their original positions, which effectively performs rationalization (trading off constraint sat-

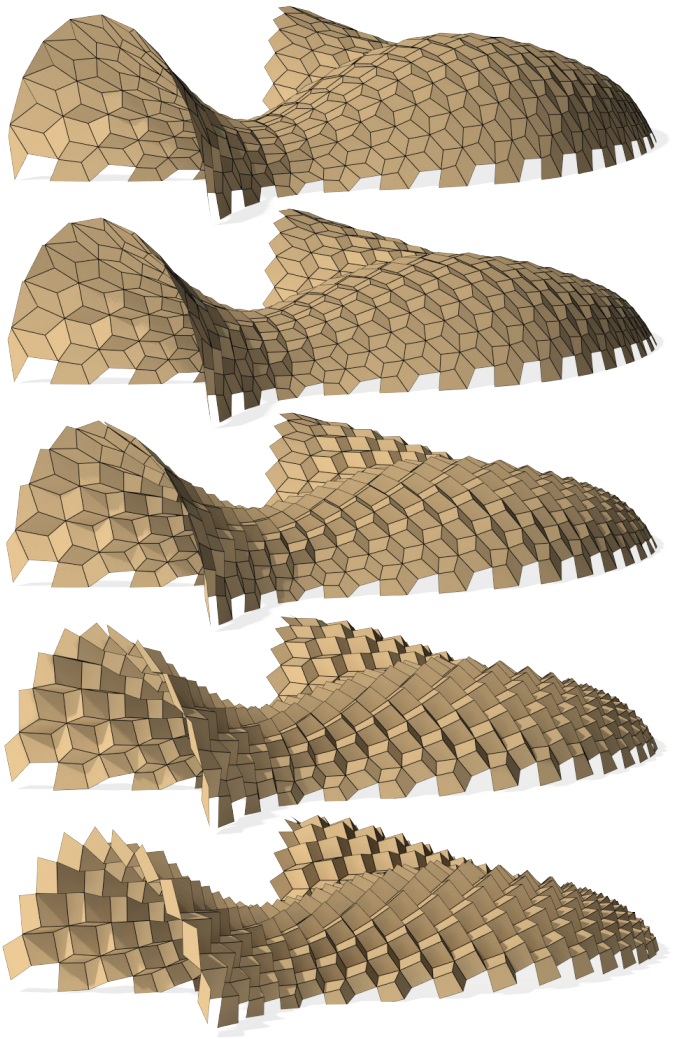

Figure 10: Shape exploration with varying weights of soft constraints. The given mesh is a quad mesh with valence three for all interior vertices. The exploration is done under hard constraints for bounded diagonal distance of faces, and soft constraints for regular polygonal faces. With increasing weights of soft constraints (from top to bottom: 0, 16, 100, 400, 1600) and fixed weights for other terms, the soft constraints gradually induce a cube-like pattern across the surface.

isfaction against deviation from the original surface). In the exploration phase, the user is allowed to change their constraint specification (by adding or removing constraints, changing weights, etc.) on the fly, to switch between different shape spaces.

To achieve interactive exploration, it is crucial that we solve the optimization

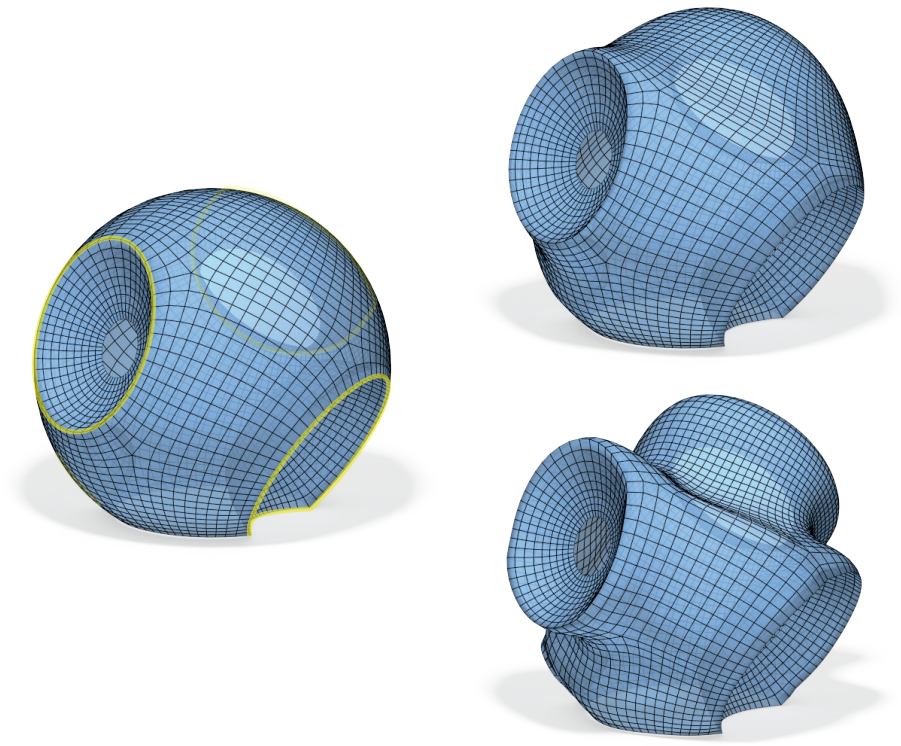

Figure 11: Shape exploration of a planar quad mesh under different extra shape constraints. Upper left: initial PQ mesh, with the handle vertices highlighted in yellow. Upper-right: result under hard constraints of bounded edge length. Bottom-left: result under hard constraints of bounded edge length and soft constraints of regular polygonal faces.

problem efficiently. Since the independent subproblems of our augmented Lagrangian solver enables parallel solving, we implement it on GPU using CUDA. Interested readers are referred to [36] for more details. Using an NVIDIA GeForce GTX 580 graphics card, we achieve interactive performance for problems with 20K vertices and 80K auxiliary variables (see the accompanying video).

5. Results

This section presents some examples of shape exploration using our method, as well as comparison between our method and other optimization methods.

5.1. Examples

Figure 2 is a comparison between soft and hard constraints of planar faces, which shows that the our solver is able to produce results that strictly satisfy hard constraints.

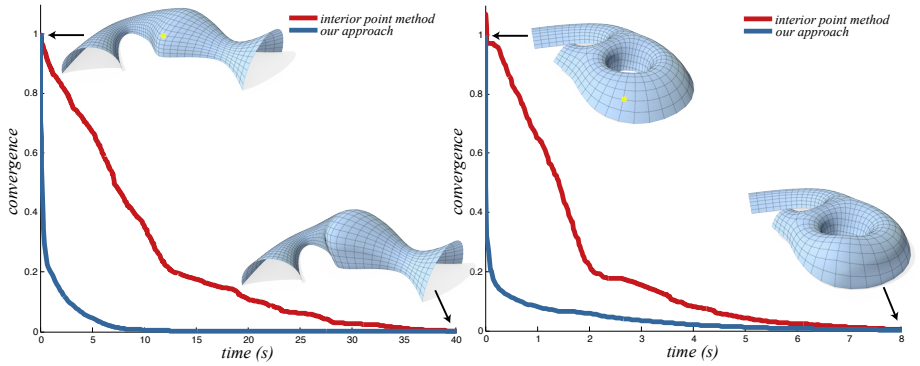

Figure 12: Comparison of convergence speed between our method and the interior point method. Each method is used to solve the constrained optimization problems for deforming two quad meshes subject to hard constraints of bounded diagonal distance. For each mesh, all boundary vertices and one interior vertex are used as handles. For all boundary handle vertices, the target positions are the same as their initial positions. In each example, the upper left image illustrates the initial mesh shape, with the interior handle vertex shown in yellow; the lower right image is the optimization solution. The plots of deviation function $\delta(t)$ (defined in Equation (12)) show that our method converges rapidly to an approximate solution.

In Figure 5 we compare between the hard constraints of planar faces and bounded diagonal distance on the same quad mesh, showing that bounded diagonal distance provides more degrees of freedom to achieve smoother shapes. In this paper, we always set the diagonal distance tolerance to 1% of the average edge length of the initial mesh, according to [2].

Figures 1 and 9 are examples of shape exploration by dragging handles. Figure 1 is a tri-hex mesh subject to a hard constraint of face planarity and a soft constraint of regular polygonal faces, while Figure 9 is a quad mesh subject to a hard constraint of bounded diagonal distance. The exploration sessions of these two models are also shown in the accompanying video, from which we can see the interactive performance of the system.

Figures 8, 11 and 10 show shape exploration for a mesh with different constraint specifications. Figures 8 and 11 are examples with different constraint types on same meshes, while Figure 10 is an example under a same set of constraints but different soft constraint weights.

5.2. Comparison

To evaluate the efficiency of our method, Figure 12 compares the performance between our method and the interior point method [37], which is a popular choice for general constrained optimization problems. The interior point solver is implemented using the commercial software package KNITRO [38]. Since KNITRO is only available on CPU, we compare it against a CPU implementation of our method. To speed up the interior point solver, we evaluate the Jacobian of the constraint functions in parallel using OpenMP. The two

methods are applied to the same optimization problems for handle-based deformation of constrained meshes, and converge to the same results. To compare their convergence speed, Figure 12 plots for each method,

$$\delta(t) = \frac{\|\mathbf{p}(t) - \mathbf{p}^*\|}{\|\mathbf{p}_0 - \mathbf{p}^*\|}, \tag{12}$$

indicating the deviation between intermediate mesh shapes during optimization and the final shape, where $\mathbf{p}(t)$ is the mesh shape at time instance t, and $\mathbf{p}_0$, $\mathbf{p}^*$ are the initial and final mesh shapes respectively. Our method decreases $\delta(t)$ rapidly at the start. When a user drags the vertex handles continuously, it is important to update the mesh shape promptly to provide visual feedback to the user. An approximate solution is sufficient for intuitive feedback, and our method is well suited for this scenario. With a GPU implementation, the decrease of $\delta(t)$ for our method is even faster (see [36]), enabling realtime deformation of constrained meshes.

6. Discussion and Conclusion

A key factor for the efficiency of our method is the use of auxiliary variables, together with the alternating minimization strategy for the primal update. First of all, this leads to separable subproblems for updating auxiliary variables, which gains speedup easily from parallelism. Moreover, the update of vertex variables only requires solving a linear system of fixed size, regardless of the number of constraints of the problem. Additionally, the linear systems are predefined, making it possible to prefactorize them to accelerate the solving. On the contrary, the interior point method needs to solve a linear system involving both the primal and the dual variables at each iteration. Increasing the number of constraints leads to larger linear systems and thus more time needed for each solve. Furthermore, each iteration solves a different linear system, making it difficult to gain speedup from numerical prefactorization. The fast convergence of our method at the beginning of the optimization is typical for numerical solvers that employ auxiliary variables and proximal operators [32]. On the other hand, the final convergence of such solvers is slow (typically linear), and it might take a long time to converge to a solution of high accuracy [39]. In the future, in order to improve the final convergence rate, we can either employ a hybrid approach which switches to Newton-type solvers at the final stage, or speed up the dual problem using accelerated gradient descent similar to [40].

Another limitation of our method is that it cannot handle arbitrary hard constraints. Hard constraints are often desirable in the context of architectural geometry and physical production. It can be tempting for a user to set a large number of constraints as hard constraints. However, if the number of hard constraints is too large, the problem may not have a feasible solution. In this case, our optimization may become unstable and the output may not satisfy any of the prescribed constraints. In future work, we would like to explore heuristics and algorithms to detect if the problem is not feasible and to find a minimal

set of constraints to remove in order to make the problem feasible. Another drawback of this method of optimization is that it is not easy to predict how changing the weights for soft constraints influences the final model. An avenue of future work is to perform sensitivity analysis on the weights and give guidance to the user whether small or large changes can have significant effect on the final result.

To conclude, we have presented a framework for interactive shape exploration of architectural designs based on shape constraints. Our new formulation provides a simple, but effective recipe to enforce soft and hard constraints while allowing interactive performances. Mixing soft and hard constraints during the exploration phase helps to achieve designs that are both aesthetically pleasing and respect constraints imposed by material, fabrication, or assembly.

Acknowledgements

The models are provided by Yang Liu (Figure 2), Asymptote Architecture and Waagner Biro (Figure 8), Zaha Hadid Architects and Amir Vaxman (Figure 1). The model in Figure 12 right is taken from [13]. This work has been supported by Swiss National Science Foundation (SNSF) grants 20PA21L_129607 and 200021_137626. This research has received funding from the European Research Council under the European Unions Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement 257453, ERC Starting Grant COSYM.

References

[1] Glymph, J., Shelden, D., Ceccato, C., Mussel, J., Schober, H.. A parametric strategy for free-form glass structures using quadrilateral planar facets. Automation in Construction 2004;13(2):187 – 202.

[2] Yang, Y.L., Yang, Y.J., Pottmann, H., Mitra, N.J.. Shape space exploration of constrained meshes. ACM Trans Graph 2011;30(6):124:1–124:12.

[3] Ceccato, C., Hesselgren, L., Pauly, M., Pottmann, H., Wallner, J., editors. Advances in Architectural Geometry 2010. Springer; 2010.

[4] Hesselgren, L., Sharma, S., Wallner, J., Baldassini, N., Bompas, P., Raynaud, J., editors. Advances in Architectural Geometry 2012. Springer; 2013.

[5] Liu, Y., Pottmann, H., Wallner, J., Yang, Y.L., Wang, W.. Geometric modeling with conical meshes and developable surfaces. ACM Trans Graph 2006;25(3):681–689.

[6] Wang, W., Liu, Y., Yan, D.M., Chan, B., Ling, R., Sun, F.. Hexagonal meshes with planar faces. Tech. Rep.; Department of Computer Science, The University of Hong Kong; 2008.

[7] Wang, W., Liu, Y.. A note on planar hexagonal meshes. In: Nonlinear Computational Geometry; vol. 151 of *The IMA Volumes in Mathematics and its Applications*. 2010, p. 221–233.

[8] Zadravec, M., Schiftner, A., Wallner, J.. Designing quad-dominant meshes with planar faces. Comput Graph Forum 2010;29(5):1671–1679.

[9] Schiftner, A., Balzer, J.. Statics-sensitive layout of planar quadrilateral meshes. In: Advances in Architectural Geometry 2010. 2010, p. 221–236.

[10] Liu, Y., Xu, W., Wang, J., Zhu, L., Guo, B., Chen, F., et al. General planar quadrilateral mesh design using conjugate direction field. ACM Trans Graph 2011;30(6).

[11] Zimmer, H., Campen, M., Herkrath, R., Kobbelt, L.. Variational tangent plane intersection for planar polygonal meshing. In: Advances in Architectural Geometry 2012. 2013, p. 319–332.

[12] Pottmann, H., Liu, Y., Wallner, J., Bobenko, A., Wang, W.. Geometry of multi-layer freeform structures for architecture. ACM Trans Graph 2007;26(3).

[13] Pottmann, H., Schiftner, A., Bo, P., Schmiedhofer, H., Wang, W., Baldassini, N., et al. Freeform surfaces from single curved panels. ACM Trans Graph 2008;27(3):76:1–76:10.

[14] Pottmann, H., Huang, Q., Deng, B., Schiftner, A., Kilian, M., Guibas, L., et al. Geodesic patterns. ACM Trans Graph 2010;29.

[15] Wallner, J., Schiftner, A., Kilian, M., Flry, S., Hbinger, M., Deng, B., et al. Tiling freeform shapes with straight panels: Algorithmic methods. In: Advances in Architectural Geometry 2010. 2010, p. 73–86.

[16] Eigensatz, M., Kilian, M., Schiftner, A., Mitra, N.J., Pottmann, H., Pauly, M.. Paneling architectural freeform surfaces. ACM Trans Graph 2010;29(4):45:1–45:10.

[17] Eigensatz, M., Deuss, M., Schiftner, A., Kilian, M., Mitra, N.J., Pottmann, H., et al. Case studies in cost-optimized paneling of architectural freeform surfaces. In: Advances in Architectural Geometry. 2010,.

[18] Flöry, S., Pottmann, H.. Ruled surfaces for rationalization and design in architecture. In: LIFE in:formation. On Responsive Information and Variations in Architecture. 2010, p. 103–109. Proc. ACADIA 2010.

[19] Flöry, S., Nagai, Y., Isvoranu, F., Pottmann, H., Wallner, J.. Ruled free forms. In: Advances in Architectural Geometry 2012. 2013, p. 57–66.

[20] Schiftner, A., Höbinger, M., Wallner, J., Pottmann, H.. Packing circles and spheres on surfaces. ACM Trans Graphics 2009;28(5).

[21] Bo, P., Pottmann, H., Kilian, M., Wang, W., Wallner, J.. Circular arc structures. ACM Trans Graphics 2011;30(4).

[22] Zimmer, H., Campen, M., Bommes, D., Kobbelt, L.. Rationalization of triangle-based point-folding structures. Comp Graph Forum 2012;31(2pt3):611–620.

[23] Deng, B., Pottmann, H., Wallner, J.. Functional webs for freeform architecture. Comput Graph Forum 2011;30(5):1369–1378.

[24] Kilian, M., Mitra, N.J., Pottmann, H.. Geometric modeling in shape space. ACM Trans Graph 2007;26(3).

[25] Zhao, X., Tang, C.C., Yang, Y.L., Pottmann, H., Mitra, N.J.. Intuitive design exploration of constrained meshes. In: Advances in Architectural Geometry 2012. 2013, p. 305–318.

[26] Vaxman, A.. Modeling polyhedral meshes with affine maps. Computer Graphics Forum 2012;31(5):1647–1656.

[27] Poranne, R., Chen, R., Gotsman, C.. On Linear Spaces of Polyhedral Meshes. ArXiv e-prints 2013;1303.4110.

[28] Bouaziz, S., Deuss, M., Schwartzburg, Y., Weise, T., Pauly, M.. Shape-up: Shaping discrete geometry with projections. Computer Graphics Forum 2012;31(5):1657–1667.

[29] Poranne, R., Ovreiu, E., Gotsman, C.. Interactive planarization and optimization of 3d meshes. Computer Graphics Forum 2013;32(1):152–163.

[30] Deng, B., Bouaziz, S., Deuss, M., Zhang, J., Schwartzburg, Y., Pauly, M.. Exploring local modifications for constrained meshes. Computer Graphics Forum (Proceedings of Eurographics 2013) 2013;32(2):11–20.

[31] Combettes, P.L., Pesquet, J.C.. Proximal splitting methods in signal processing. In: Fixed-Point Algorithms for Inverse Problems in Science and Engineering. Springer; 2011,.

[32] Parikh, N., Boyd, S.. Proximal algirhtms. Foundations and Trends in Optimization 2014;1(3):123–231.

[33] Bertsekas, D.P.. Constrained Optimization and Lagrange Multiplier Methods. Athena Scientific; 1996.

[34] Eckstein, J.. Splitting methods for monotone operators with applications to parallel optimization. Ph.D. thesis; MIT; 1989.

[35] Golub, G., Loan, C.V.. Matrix Computations. Johns Hopkins University Press; 3rd ed.; 1996.

[36] Kaspar, A., Deng, B.. Realtime deformation of constrained meshes using gpu. In: 2013 Symposium on GPU Computing and Applications. 2013,.

[37] Nocedal, J., Wright, S.J.. Numerical Optimization. Springer; 2nd ed.; 2006.

[38] Byrd, R.H., Nocedal, J., Waltz, R.A.. KNITRO: An integrated package for nonlinear optimization. In: Pillo, G., Roma, M., editors. Large-Scale Nonlinear Optimization; vol. 83 of *Nonconvex Optimization and Its Applications*. Springer US; 2006, p. 35–59.

[39] Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.. Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends in Machine Learning 2011;3(1):1–122.

[40] Goldstein, T., O'Donoghue, B., Setzer, S., Baraniuk, R.. Fast alternating direction optimization methods. UCLA CAM Reports; 2012.